Object Removal by Exemplar-Based Inpainting

A. Criminisi, P. Pérez
Microsoft Research Ltd., Cambridge, UK
antcrim@microsoft.com, pperez@microsoft.com

K. Toyama
Microsoft Corporation, Redmond, WA, USA
kentoy@microsoft.com

Abstract

A new algorithm is proposed for removing large objects from digital images. The challenge is to fill in the hole that is left behind in a visually plausible way.

In the past, this problem has been addressed by two classes of algorithms: (i) "texture synthesis" algorithms for generating large *image regions from sample textures, and (ii) "inpainting" techniques for filling in* small *image gaps. The former work well for "textures" – repeating two-dimensional patterns with some stochasticity; the latter focus on linear "structures" which can be thought of as one-dimensional patterns, such as lines and object contours.*

This paper presents a novel and efficient algorithm that combines the advantages of these two approaches. We first note that exemplar-based texture synthesis contains the essential process required to replicate both texture and structure; the success of structure propagation, however, is highly dependent on the order in which the filling proceeds. We propose a best-first algorithm in which the confidence *in the synthesized pixel values is propagated in a manner similar to the propagation of information in inpainting. The actual colour values are computed using exemplar-based synthesis. Computational efficiency is achieved by a block-based sampling process.*

A number of examples on real and synthetic images demonstrate the effectiveness of our algorithm in removing large occluding objects as well as thin scratches. Robustness with respect to the shape of the manually selected target region is also demonstrated. Our results compare favorably to those obtained by existing techniques.

1. Introduction

This paper presents a novel algorithm for removing objects from digital photographs and replacing them with visually plausible backgrounds. Figure 1 shows an example of this task, where the foreground person (manually selected as the *target region*) is replaced by textures sampled from the remainder of the image. The algorithm effectively hallucinates new colour values for the target region in a way that looks "reasonable" to the human eye.

In previous work, several researchers have considered texture synthesis as a way to fill large image regions with

a **b**

Figure 1: **Removing large objects from images.** (a) Original image. (b) The region corresponding to the foreground person (covering about 19% of the image) has been manually selected and then automatically removed. Notice that the horizontal structures of the fountain have been synthesized in the occluded area together with the water, grass and rock textures.

"pure" textures – repetitive two-dimensional textural patterns with moderate stochasticity. This is based on a large body of texture-synthesis research, which seeks to replicate texture *ad infinitum*, given a small source sample of pure texture [1, 8, 9, 10, 11, 12, 14, 15, 16, 19, 22]. Of particular interest are *exemplar-based techniques* which cheaply and effectively generate new texture by sampling and copying colour values from the source [1, 9, 10, 11, 15].

As effective as these techniques are in replicating consistent texture, they have difficulty filling holes in photographs of real-world scenes, which often consist of linear structures and *composite textures* – multiple textures interacting spatially [23]. The main problem is that boundaries between image regions are a complex product of mutual influences between different textures. In constrast to the two-dimensional nature of pure textures, these boundaries form what might be considered more one-dimensional, or linear, image structures.

A number of algorithms specifically address this issue for the task of image restoration, where speckles, scratches, and overlaid text are removed [2, 3, 4, 7, 20]. These *image inpainting* techniques fill holes in images by propagating linear structures (called *isophotes* in the inpainting literature) into the target region via diffusion. They are inspired by the partial differential equations of physical heat flow,

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

and work convincingly as restoration algorithms. Their drawback is that the diffusion process introduces some blur, which is noticeable when the algorithm is applied to fill larger regions.

The algorithm presented here combines the strengths of both approaches. As with inpainting, we pay special attention to linear structures. But, linear structures abutting the target region only influence the fill order of what is at core an exemplar-based texture synthesis algorithm. The result is an algorithm that has the efficiency and qualitative performance of exemplar-based texture synthesis, but which also respects the image constraints imposed by surrounding linear structures.

Our algorithm builds on very recent research along similar lines. The work in [5] decomposes the original image into two components; one of which is processed by inpainting and the other by texture synthesis. The output image is the sum of the two processed components. This approach still remains limited to the removal of small image gaps, however, as the diffusion process continues to blur the filled region (*cf.*, [5], fig.5 top right). The automatic switching between "pure texture-" and "pure structure-mode" described in [21] is also avoided.

One of the first attempts to use exemplar-based synthesis specifically for object removal was by Harrison [13]. There, the order in which a pixel in the target region is filled was dictated by the level of "texturedness" of the pixel's neighborhood[1]. Although the intuition is sound, strong linear structures were often overruled by nearby noise, minimizing the value of the extra computation. A related technique drove the fill order by the local shape of the target region, but did not seek to explicitly propagate linear structure [6].

Finally, Zalesny et al. [23] describe an interesting algorithm for the parallel synthesis of composite textures. They devise a special-purpose solution for the interface between two textures. In this paper we show that, in fact, only one mechanism is sufficient for the synthesis of both pure and composite textures.

Section 2 presents the key observation on which our algorithm depends. Section 3 describes the details of the algorithm. Results on both synthetic and real imagery are presented in section 4.

2. Exemplar-based synthesis suffices

The core of our algorithm is an isophote-driven image-sampling process. It is well-understood that exemplar-based approaches perform well for two-dimensional textures [1, 9, 15]. But, we note in addition that exemplar-based texture synthesis is sufficient for propagating extended linear image structures, as well. A separate synthesis mechanism is not required for handling isophotes.

[1] An implementation of Harrison's algorithm is available from `www.csse.monash.edu.au/~pfh/resynthesizer/`

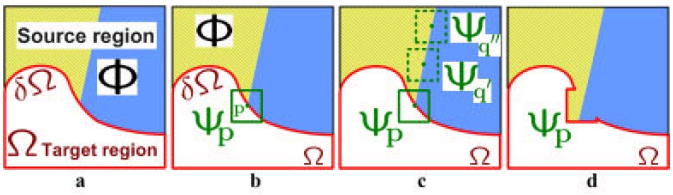


Figure 2: **Structure propagation by exemplar-based texture synthesis.** (a) Original image, with the *target region* Ω, its contour $\delta\Omega$ and the *source region* Φ clearly marked. (b) We want to synthesize the area delimited by the patch $\Psi_{\mathbf{p}}$ centred on the point $\mathbf{p} \in \delta\Omega$. (c) The most likely candidate matches for $\Psi_{\mathbf{p}}$ lie along the boundary between the two textures in the source region, *e.g.*, $\Psi_{\mathbf{q}'}$ and $\Psi_{\mathbf{q}''}$. (d) The best matching patch in the candidates set has been copied into the position occupied by $\Psi_{\mathbf{p}}$, thus achieving partial filling of Ω. The target region Ω has, now, shrank and its front has assumed a different shape. See text for details.

Figure 2 illustrates this point. For ease of comparison, we adopt notation similar to that used in the inpainting literature. The region to be filled, *i.e.*, the *target* region is indicated by Ω, and its contour is denoted $\delta\Omega$. The contour evolves inward as the algorithm progresses, and so we also refer to it as the "fill front". The *source* region, Φ, which remains fixed throughout the algorithm, provides samples used in the filling process.

We now focus on a single iteration of the algorithm to show how structure and texture are adequately handled by exemplar-based synthesis. Suppose that the square template $\Psi_{\mathbf{p}} \in \Omega$ centred at the point $\mathbf{p}$ (fig. 2b), is to be filled. The best-match sample from the source region comes from the patch $\Psi_{\hat{\mathbf{q}}} \in \Phi$, which is most similar to those parts that are already filled in $\Psi_{\mathbf{p}}$. In the example in fig. 2b, we see that if $\Psi_{\mathbf{p}}$ lies on the continuation of an image edge, the most likely best matches will lie along the same (or a similarly coloured) edge (*e.g.*, $\Psi_{\mathbf{q}'}$ and $\Psi_{\mathbf{q}''}$ in fig. 2c).

All that is required to propagate the isophote inwards is a simple transfer of the pattern from the best-match source patch (fig. 2d). Notice that isophote orientation is automatically preserved. In the figure, despite the fact that the original edge is not orthogonal to the target contour $\delta\Omega$, the propagated structure has maintained the same orientation as in the source region.

3. Region-filling algorithm

We now proceed with the details of our algorithm.

First, a user selects a target region, Ω, to be removed and filled. The source region, Φ, may be defined as the entire image minus the target region ($\Phi = \mathcal{I} - \Omega$), as a dilated band around the target region, or it may be manually specified by the user.

Next, as with all exemplar-based texture synthesis [10], the size of the template window Ψ must be specified. We provide a default window size of 9×9 pixels, but in practice require the user to set it to be slightly larger than the largest

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

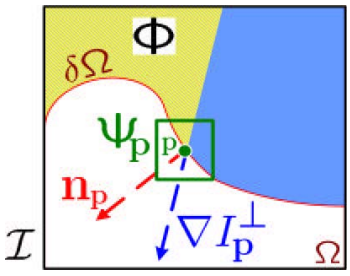


Figure 3: **Notation diagram.** Given the patch $\Psi_{\mathbf{p}}$, $\mathbf{n}_{\mathbf{p}}$ is the normal to the contour $\delta\Omega$ of the target region Ω and $\nabla I_{\mathbf{p}}^{\perp}$ is the isophote (direction and intensity) at point $\mathbf{p}$. The entire image is denoted with $\mathcal{I}$.

distinguishable texture element, or "texel", in the source region.

Once these parameters are determined, the remainder of the region-filling process is completely automatic.

In our algorithm, each pixel maintains a *colour* value (or "empty", if the pixel is unfilled) and a *confidence* value, which reflects our confidence in the pixel value, and which is frozen once a pixel has been filled. During the course of the algorithm, patches along the fill front are also given a temporary *priority* value, which determines the order in which they are filled. Then, our algorithm iterates the following three steps until all pixels have been filled:

1. Computing patch priorities. Filling order is crucial to non-parametric texture synthesis [1, 6, 10, 13]. Thus far, the default favourite has been the "onion peel" method, where the target region is synthesized from the outside inward, in concentric layers. To our knowledge, however, designing a fill order which explicitly encourages propagation of linear structure (together with texture) has never been explored.

Our algorithm performs this task through a best-first filling algorithm that depends entirely on the priority values that are assigned to each patch on the fill front. The priority computation is biased toward those patches which are on the continuation of strong edges and which are surrounded by high-confidence pixels.

Given a patch $\Psi_{\mathbf{p}}$ centred at the point $\mathbf{p}$ for some $\mathbf{p} \in \delta\Omega$ (see fig. 3), its priority $P(\mathbf{p})$ is defined as the product of two terms:

$$P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p}). \tag{1}$$

We call $C(\mathbf{p})$ the *confidence* term and $D(\mathbf{p})$ the *data* term, and they are defined as follows:

$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Psi_{\mathbf{p}}\cap\bar{\Omega}} C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}, \qquad D(\mathbf{p}) = \frac{|\nabla I_{\mathbf{p}}^{\perp} \cdot \mathbf{n}_{\mathbf{p}}|}{\alpha}$$

where $|\Psi_{\mathbf{p}}|$ is the area of $\Psi_{\mathbf{p}}$, α is a normalization factor (*e.g.*, $\alpha = 255$ for a typical grey-level image), and $\mathbf{n}_{\mathbf{p}}$ is a unit vector orthogonal to the front $\delta\Omega$ in the point $\mathbf{p}$. The priority is computed for every border patch, with distinct patches for each pixel on the boundary of the target region.

During initialization, the function $C(\mathbf{p})$ is set to $C(\mathbf{p}) = 0 \ \ \forall \mathbf{p} \in \Omega$, and $C(\mathbf{p}) = 1 \ \ \forall \mathbf{p} \in \mathcal{I} - \Omega$.

The confidence term $C(\mathbf{p})$ may be thought of as a measure of the amount of reliable information surrounding the pixel $\mathbf{p}$. The intention is to fill first those patches which have more of their pixels already filled, with additional preference given to pixels that were filled early on (or that were never part of the target region).

This automatically incorporates preference towards certain shapes along the fill front. For example, patches that include corners and thin tendrils of the target region will tend to be filled first, as they are surrounded by more pixels from the original image. These patches provide more reliable information against which to match. Conversely, patches at the tip of "peninsulas" of filled pixels jutting into the target region will tend to be set aside until more of the surrounding pixels are filled in.

At a coarse level, the term $C(\mathbf{p})$ of (1) approximately enforces the desirable concentric fill order. As filling proceeds, pixels in the outer layers of the target region will tend to be characterized by greater confidence values, and therefore be filled earlier; pixels in the centre of the target region will have lesser confidence values.

The data term $D(\mathbf{p})$ is a function of the strength of isophotes hitting the front $\delta\Omega$ at each iteration. This term boosts the priority of a patch that an isophote "flows" into. This factor is of fundamental importance in our algorithm because it encourages linear structures to be synthesized first, and, therefore propagated securely into the target region. Broken lines tend to connect, thus realizing the "Connectivity Principle" of vision psychology [7, 17] (*cf.*, fig. 4, fig. 7d, fig. 8b and fig. 13d).

There is a delicate balance between the confidence and data terms. The data term tends to push isophotes rapidly inward, while the confidence term tends to suppress precisely this sort of incursion into the target region. As presented in the results section, this balance is handled gracefully via the mechanism of a single priority computation for all patches on the fill front.

Since the fill order of the target region is dictated solely by the priority function $P(\mathbf{p})$, we avoid having to predefine an arbitrary fill order as done in existing patch-based approaches [9, 19]. Our fill order is function of image properties, resulting in an organic synthesis process that eliminates the risk of "broken-structure" artefacts (fig. 7c) and also reduces blocky artefacts without an expensive patch-cutting step [9] or a blur-inducing blending step [19].

2. Propagating texture and structure information. Once all priorities on the fill front have been computed, the patch $\Psi_{\hat{\mathbf{p}}}$ with highest priority is found. We then fill it with data extracted from the source region Φ.

In traditional inpainting techniques, pixel-value informa-

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

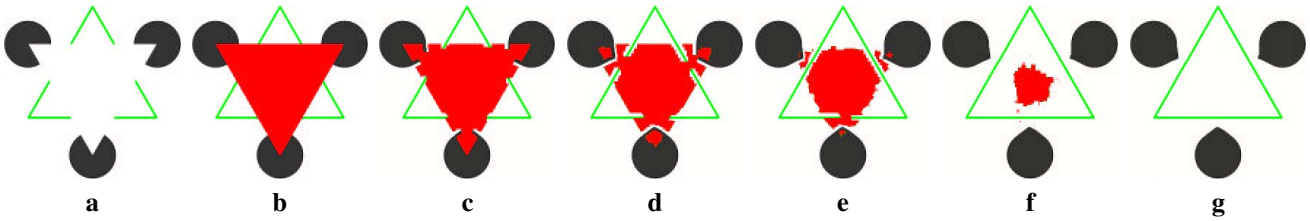


Figure 4: **Realization of the "Connectivity Principle" [7, 17] on a synthetic example.** (a) Original image, the "Kanizsa triangle" with random noise added. (b) The occluding white triangle in the original image has been manually selected as the target region (24% of total image area) and marked in red. (c. . .f) Different stages of the filling process. (d) Notice that strong edges are pushed inside the target region first and that sharp appendices (*e.g.*, the vertices of the red triangle) are rapidly smoothed. (f) When no structures hit the front $\delta\Omega$ the target region evolves in a roughly circular shape. (g) The output image where the target region has been filled, *i.e.*, the occluding triangle removed. Little imperfections are present in the curvature of the circles in the reconstructed areas, while the sides of the internal triangle have been correctly connected. The blur typical of diffusion techniques is completely avoided. See figs. 7, 8, 13 for further examples of structural continuation.

tion is propagated via diffusion. As noted previously, diffusion necessarily leads to image smoothing, which results in blurry fill-in, especially of large regions (see fig. 10f).

On the contrary, we propagate image texture by direct sampling of the source region. Similar to [10], we search in the source region for that patch which is most similar to $\Psi_{\hat{\mathbf{p}}}$.[2] Formally,

$$\Psi_{\hat{\mathbf{q}}} = \arg\min_{\Psi_{\mathbf{q}} \in \Phi} d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}}) \qquad (2)$$

where the distance $d(\Psi_{\mathbf{a}}, \Psi_{\mathbf{b}})$ between two generic patches $\Psi_{\mathbf{a}}$ and $\Psi_{\mathbf{b}}$ is simply defined as the sum of squared differences (SSD) of the already filled pixels in the two patches. We use the `CIE Lab` colour space because of its property of perceptual uniformity [18].

Having found the source *exemplar* $\Psi_{\hat{\mathbf{q}}}$, the value of each pixel-to-be-filled, $\mathbf{p}' \;|\mathbf{p}' \in \Psi_{\hat{\mathbf{p}} \cap \Omega}$, is copied from its corresponding position inside $\Psi_{\hat{\mathbf{q}}}$.

This suffices to achieve the propagation of both structure and texture information from the source Φ to the target region Ω, one patch at a time (*cf.*, fig. 2d). In fact, we note that any further manipulation of the pixel values (*e.g.*, adding noise, smoothing and so forth) that does not explicitly depend upon statistics of the source region, is far more likely to degrade visual similarity between the filled region and the source region, than to improve it.

3. Updating confidence values. After the patch $\Psi_{\hat{\mathbf{p}}}$ has been filled with new pixel values, the confidence $C(\mathbf{p})$ is updated in the area delimited by $\Psi_{\hat{\mathbf{p}}}$ as follows:

$$C(\mathbf{q}) = C(\hat{\mathbf{p}}) \quad \forall \mathbf{q} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega.$$

This simple update rule allows us to measure the relative confidence of patches on the fill front, without image-specific parameters. As filling proceeds, confidence values decay, indicating that we are less sure of the colour values of pixels near the centre of the target region.

[2] Valid patches $\Psi_{\hat{\mathbf{q}}}$ must be entirely contained in Φ.

- Extract the manually selected initial front $\delta\Omega^0$.
- Repeat until done:
  - **1a.** Identify the fill front $\delta\Omega^t$. If $\Omega^t = \emptyset$, exit.
  - **1b.** Compute priorities $P(\mathbf{p}) \quad \forall \mathbf{p} \in \delta\Omega^t$.
  - **2a.** Find the patch $\Psi_{\hat{\mathbf{p}}}$ with the maximum priority, *i.e.*, $\Psi_{\hat{\mathbf{p}}} \mid \hat{\mathbf{p}} = \arg\max_{\mathbf{p} \in \delta\Omega^t} P(\mathbf{p})$
  - **2b.** Find the exemplar $\Psi_{\hat{\mathbf{q}}} \in \Phi$ that minimizes $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$.
  - **2c.** Copy image data from $\Psi_{\hat{\mathbf{q}}}$ to $\Psi_{\hat{\mathbf{p}}}$.
  - **3.** Update $C(\mathbf{p}) \quad \forall \mathbf{p} \;|\mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$

Table 1: **Region filling algorithm.**

A pseudo-code description of the algorithmic steps is shown in table 1. The superscript t indicates the current iteration.

4. Results and comparisons

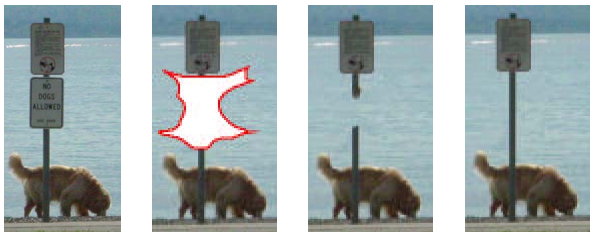
Here we apply our algorithm to a variety of images, ranging from purely synthetic images to full-colour photographs that include complex textures. Where possible, we make side-by-side comparisons to previously proposed methods. In other cases, we hope the reader will refer to the original source of our test images (many are taken from previous literature on inpainting and texture synthesis) and compare these results with the results of earlier work.

In all of the experiments, the patch size was set to be greater than the largest texel or the thickest structure (*e.g.*, edges) in the source region. Furthermore, unless otherwise stated the source region has been set to be $\Phi = \mathcal{I} - \Omega$. All experiments were run on a 2.5GHz Pentium IV with 1GB of RAM.

The Kanizsa triangle. We perform our first experiment on the well-known Kanizsa triangle [17] to show how the algorithm works on a structure-rich synthetic image.

As shown in fig. 4, our algorithm deforms the fill front $\delta\Omega$ under the action of two forces: isophote continuation (the data term, $D(\mathbf{p})$) and the "pressure" from surrounding filled pixels (the confidence term, $C(\mathbf{p})$).

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE



a b c d

Figure 7: **Onion peel vs. structure-guided filling.** (a) Original image. (b) The target region has been selected and marked in red. (c) Results of filling by concentric layers. (d) Results of filling with our algorithm. Thanks to the *data term* in (1) the pole is reconstructed correctly.

The sharp linear structures of the incomplete green triangle are grown into the target region. But also, no single structural element dominates all of the others; this balance among competing isophotes is achieved through the naturally decaying confidence values (in an earlier version of our algorithm which lacked this balance, "runaway" structures led to large-scale artefacts.)

Figures 4e,f also show the effect of the confidence term in smoothing sharp appendices such as the vertices of the target region (in red).

As described above, the confidence is propagated in a manner similar to the front-propagation algorithms used in inpainting. We stress, however, that unlike inpainting, it is the confidence values that are propagated along the front (and which determine fill order), not colour values themselves, which are sampled from the source region.

Finally, we note that despite the large size of the removed region, edges and lines in the filled region are as sharp as any found in the source region. There is no blurring from diffusion processes. This is a property of exemplar-based texture synthesis.

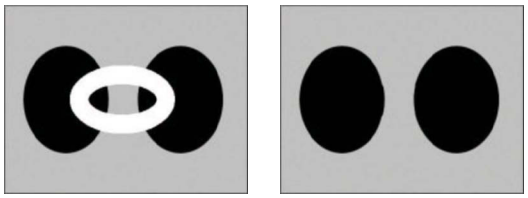
The effect of different filling strategies. Figures 5, 6 and 7 demonstrate the effect of different filling strategies.

Figure 5f shows how our filling algorithm achieves the best structural continuation in a simple, synthetic image.

Figure 6 further demonstrates the validity of our algorithm on an aerial photograph. The 40×40-pixel target region has been selected to straddle two different textures (fig. 6b). The remainder of the 200×200 image in fig. 6a was used as source for all the experiments in fig. 6.

With raster-scan synthesis (fig. 6c) not only does the top region (the river) grow into the bottom one (the city area), but visible seams also appear at the bottom of the target region. This problem is only partially addressed by a concentric filling (fig 6d). Similarly, in fig. 6e the sophisticated ordering proposed by Harrison [13] only moderately succeeds in preventing this phenomenon.

In all of these cases, the primary difficulty is that since the (eventual) texture boundary is the most constrained part



a b

Figure 8: **Comparison with traditional structure inpainting.** (a) Original image. (b) Object removal and structure recovery via our algorithm; to be compared with fig.4 in [4].

of the target region, it should be filled first. But, unless this is explicitly addressed in determining the fill order, the texture boundary is often the last part to be filled. The algorithm proposed in this paper is designed to address this problem, and thus more naturally extends the contour between the two textures as well as the vertical grey road.

In the example in fig. 6, our algorithm fills the target region in only 2 seconds, on a Pentium IV, 2.52GHz, 1GB RAM. Harrison's resynthesizer [13], which is the nearest in quality, requires approximately 45 seconds.

Figure 7 shows yet another comparison between the concentric filling strategy and the proposed algorithm. In the presence of concave target regions, the "onion peel" filling may lead to visible artefacts such as unrealistically broken structures (see the pole in fig. 7c). Conversely, the presence of the data term of (1) encourages the edges of the pole to grow "first" inside the target region and thus correctly reconstruct the complete pole (fig. 7d). This example demonstrates the robustness of the proposed algorithm with respect to the shape of the selected target region.

Comparisons with inpainting. We now turn to some examples from the inpainting literature. The first two examples show that our approach works at least as well as inpainting.

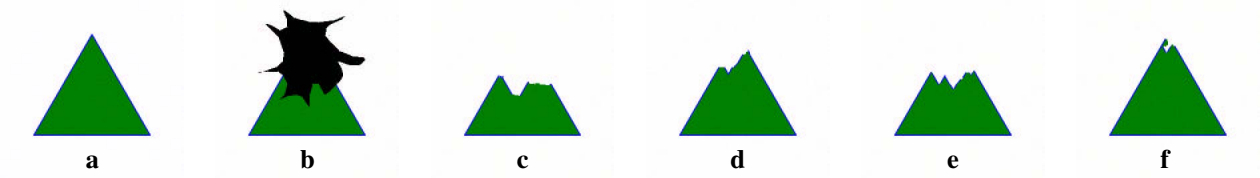
The first (fig. 8) is a synthesized image of two ellipses [4]. The occluding white torus is removed from the input image and two dark background ellipses reconstructed via our algorithm (fig. 8b). This example was chosen by authors of the original work on inpainting to illustrate the structure propagation capabilities of their algorithm. Our results are visually identical to those obtained by inpainting ([4], fig.4).

We now compare results of the restoration of an hand-drawn image. In fig. 9 the aim is to remove the foreground text. Our results (fig. 9b) are mostly indistinguishable with those obtained by traditional inpainting [3]. This example demonstrates the effectiveness of both techniques in image restoration applications.

It is in real photographs with large objects to remove,

[3] www.ece.umn.edu/users/marcelo/restoration4.html

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

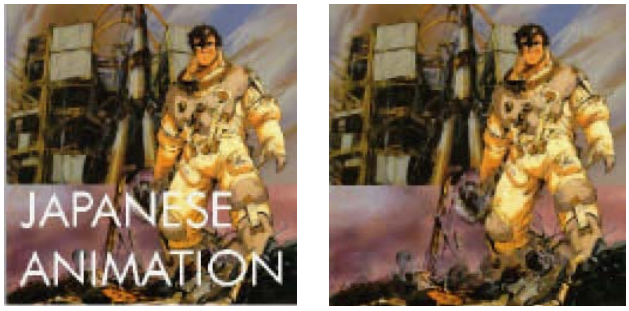


a b c d e f

Figure 5: **Effect of filling order on a synthetic image.** (a) The original image; (b) The target region has been selected and marked in black; (c) Filling the target region in raster-scan order; (d) Filling by concentric layers; (e) The result of applying Harrison's technique which took $2\ '\ 45\ ''$; (f) Filling with our algorithm which took $5\ ''$. Notice that even though the triangle upper vertex is not complete our technique performs better than the others.

a b c d e f

Figure 6: **Effect of filling order on an aerial photograph.** (a) The original image, an aerial view of London. (b) The target region has been selected and marked in red; Notice that it straddles two different textures; (c) Filling with raster-scan order; (d) Filling by concentric layers; (e) The result of applying Harrison's technique (performed in $45\ ''$); (f) Filling with our algorithm (performed in $2\ ''$). See text for details.



a b

Figure 9: **Image restoration example.** (a) Original image. The text occupies 9% of the total image area. (b) Result of text removal via our algorithm.

however, that the real advantages of our approach become apparent. Figure 10 shows an example on a real photograph, of a bungee jumper in mid-jump (from [4], fig.8). In the original work, the thin bungee cord is removed from the image via inpainting. In order to prove the capabilities of our algorithm we removed the entire bungee jumper (fig. 10e). Structures such as the shore line and the edge of the house have been automatically propagated into the target region along with plausible textures of shrubbery, water and roof tiles; and all this with no *a priori* model of anything specific to this image.

For comparison, figure 10f shows the result of filling the same target region (fig. 10b) by image inpainting[4]. Considerable blur is introduced into the target region because of inpainting's use of diffusion to propagate colour values; and high-frequency textural information is entirely absent.

[4] 120,000 iterations were run using the implementation in `www.bantha.org/~aj/inpainting/`

Figure 11 compares our algorithm to the recent "texture and structure inpainting" technique described in [5]. Figure 11(bottom right) shows that also our algorithm accomplishes the propagation of structure and texture inside the selected target region. Moreover, the lack of diffusion steps avoids blurring propagated structures (see the vertical edge in the encircled region) and makes the algorithm more computationally efficient.

Synthesizing composite textures. Fig. 12 demonstrates that our algorithm behaves well also at the boundary between two different textures, such as the ones analyzed in [23]. The target region selected in fig. 12c straddles two different textures. The quality of the "knitting" in the contour reconstructed via our approach (fig. 12d) is similar to the original image and to the results obtained in the original work (fig. 12b), but again, this has been accomplished without complicated texture models or a separate boundary-specific texture synthesis algorithm.

Further examples on photographs. We show two more examples on photographs of real scenes.

Figure 13 demonstrates, again, the advantage of the proposed approach in preventing structural artefacts (*cf.*, 7d). While the onion-peel approach produces a deformed horizon, our algorithm reconstructs the boundary between sky and sea as a convincing straight line.

Finally, in fig. 14, the foreground person has been manually selected and the corresponding region filled in automatically. The filled region in the output image convincingly mimics the complex background texture with no prominent artefacts (fig. 14f). During the filling process the topological changes of the target region are handled effortlessly.

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

Figure 10: **Removing large objects from photographs.** (a) Original image (from [4]), $205 \times 307 pix$. (b) The target region (in white) covers 12% of the total image area. (c,d) Different stages of the filling process. Notice how the isophotes hitting the boundary of the target region are propagated inwards while thin appendices (*e.g.*, the arms) in the target region tend to disappear quickly. (e) The final image where the bungee jumper has been completely removed and the occluded region reconstructed by our automatic algorithm (performed in $18\ ''$, to be compared with $10\ '$ of Harrison's resynthesizer). (f) The result of region filling by traditional image inpainting. Notice the blur introduced by the diffusion process and the complete lack of texture in the synthesized area.

5. Conclusion and future work

This paper has presented a novel algorithm for removing *large* objects from digital photographs. The result of object removal is an image in which the selected object has been replaced by a visually plausible background that mimics the appearance of the source region.

Our approach employs an exemplar-based texture synthesis technique modulated by a unified scheme for determining the fill order of the target region. Pixels maintain a confidence value, which together with image isophotes, influence their fill priority.

The technique is capable of propagating both linear structure and two-dimensional texture into the target region. Comparative experiments show that a careful selection of the fill order is necessary *and* sufficient to handle this task.

Our method performs at least as well as previous techniques designed for the restoration of small scratches, and in instances in which larger objects are removed, it dramatically outperforms earlier work in terms of both perceptual quality and computational efficiency.

Currently, we are investigating extensions for more accurate propagation of curved structures in still photographs and for object removal from video, which promise to impose an entirely new set of challenges.

Acknowledgements. The authors would like to thank M. Gangnet, A. Blake and P. Anandan for inspiring discussions; and G. Sapiro, M. Bertalmio, L. van Gool and A. Zalesny for making some of their images available.

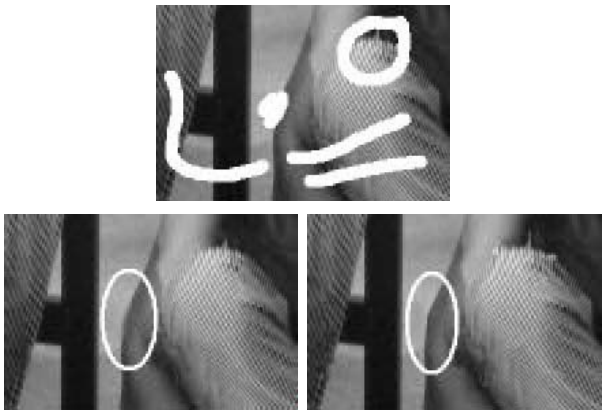


Figure 11: **Comparison with "texture and structure inpainting".** (Top) Original image (from [5]). The target regions are marked in white. (Bottom left) Region filling via the inpainting algorithm in [5]. Notice the blur of the edge in the circled region. (Bottom right) The result of our algorithm. Both structure and texture have been nicely propagated inside the target region. The edge in the circled region is noticeably sharper.

References

[1] M. Ashikhmin. Synthesizing natural textures. In *Proc. ACM Symp. on Interactive 3D Graphics*, pp. 217–226, Research Triangle Park, NC, Mar 2001.

[2] C. Ballester, V. Caselles, J. Verdera, M. Bertalmio, and G. Sapiro. A variational model for filling-in gray level and color images. In *Proc. ICCV*, pp. I: 10–16, Vancouver, Canada, Jun 2001.

[3] M. Bertalmio, A.L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proc. Conf. Comp. Vision Pattern Rec.*, pp. I:355–362, Hawai, Dec 2001.

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, pp. 417–424, New Orleans, LU, Jul 2000. `http://mountains.ece.umn.edu/~guille/inpainting.htm`.

[5] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *to appear*, 2002. `http://mountains.ece.umn.edu/~guille/inpainting.htm`.

[6] R. Bornard, E. Lecan, L. Laborelli, and J-H. Chenot. Missing data correction in still images and image sequences. In *ACM Multimedia*, France, Dec 2002.

[7] T. F. Chan and J. Shen. Non-texture inpainting by curvature-driven diffusions (CDD). *J. Visual Comm. Image Rep.*, 4(12), 2001.

[8] J.S. de Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, volume 31, pp. 361–368, 1997.

[9] A. Efros and W.T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, pp. 341–346, Eugene Fiume, Aug 2001.

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

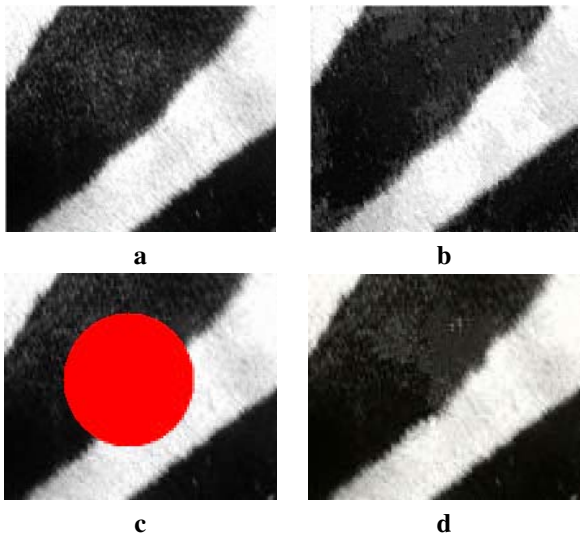


Figure 12: **Comparison with "parallel composite texture synthesis".** (a) Original image, the fur of a zebra (from [23]). (b) The result of the synthesis algorithm described in [23]. (c) Original image with the target region marked in red (22% of total image size). (d) The target region has been filled via our algorithm. The "knitting" effect along the boundary between the two textures is correctly reproduced by our technique.

[10] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, pp. 1033–1038, Kerkyra, Greece, Sep 1999.

[11] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael. Learning low-level vision. *Int. J. Computer Vision*, 40(1):25–47, 2000.

[12] D. Garber. *Computational Models for Texture Analysis and Texture Synthesis*. PhD thesis, Univ. of Southern California, USA, 1981.

[13] P. Harrison. A non-hierarchical procedure for re-synthesis of complex texture. In *Proc. Int. Conf. Central Europe Comp. Graphics, Visua. and Comp. Vision*, Plzen, Czech Republic, Feb 2001.

[14] D.J. Heeger and J.R. Bergen. Pyramid-based texture analysis/synthesis. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, volume 29, pp. 229–233, Los Angeles, CA, 1995.

[15] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, Eugene Fiume, Aug 2001.

[16] H. Igehy and L. Pereira. Image replacement through texture synthesis. In *Proc. Int. Conf. Image Processing*, pp. III:186–190, 1997.

[17] G. Kanizsa. *Organization in Vision*. Praeger, New York, 1979.

[18] J. M. Kasson and W. Plouffe. An analysis of selected computer interchange color spaces. In *ACM Transactions on Graphics*, volume 11, pp. 373–405, Oct 1992.

[19] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. In *ACM Transactions on Graphics*, 2001.

[20] S. Masnou and J.-M. Morel. Level lines based disocclusion. In *Int. Conf. Image Processing*, Chicago, 1998.

[21] S. Rane, G. Sapiro, and M. Bertalmio. Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. In *IEEE. Trans. Image Processing*, 2002. to appear.

[22] L.-W. Wey and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, 2000.

[23] A. Zalesny, V. Ferrari, G. Caenen, and L. van Gool. Parallel composite texture synthesis. In *Texture 2002 workshop - (in conjunction with ECCV02)*, Copenhagen, Denmark, Jun 2002.

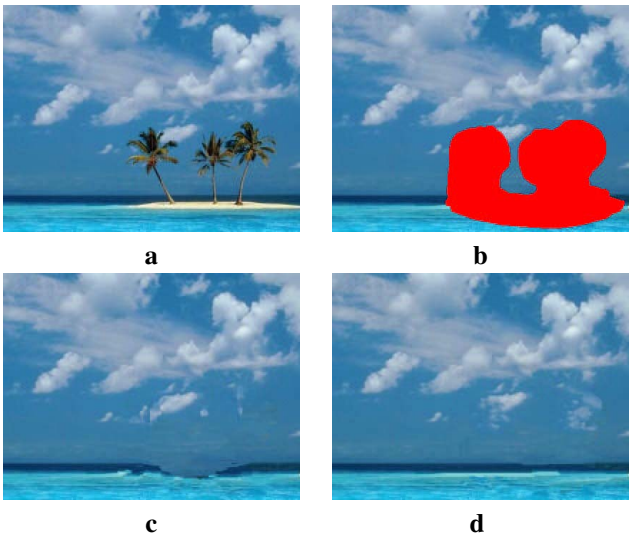


Figure 13: **Concentric-layer filling vs. the proposed guided filling algorithm.** (a) Original image. (b) The manually selected target region (20% of the total image area, in red). (c) The result of filling by concentric layers. The deformation of the horizon is caused by the fact that in the concentric layers filling sky and sea grow inwards at the same speed. Thus, the reconstructed sky-sea boundary tends to follow the *skeleton* of the selected target region. (d) The result of filling by the proposed algorithm. The horizon is correctly reconstructed as a straight line.

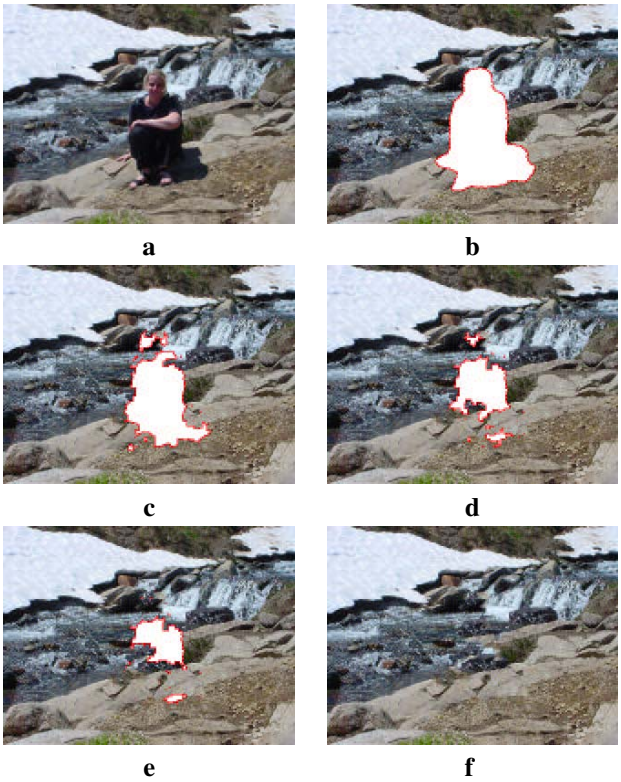


Figure 14: **Removing large objects from photographs.** (a) Original image. (b) The target region (10% of the total image area) has been blanked out. (c. . .e) Intermediate stages of the filling process. (f) The target region has been completely filled and the selected object removed. The source region has been automatically selected as a band around the target region. The edges of the stones have been nicely propagated inside the target region together with the water texture.

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE